

# PROGRAMAÇÃO COMPUTACIONAL PARA ENGENHARIA

## ESTRUTURA DE REPETIÇÃO

Maurício Moreira Neto<sup>1</sup>

<sup>1</sup> **Universidade Federal do Ceará**  
**Departamento de Computação**

30 de janeiro de 2020

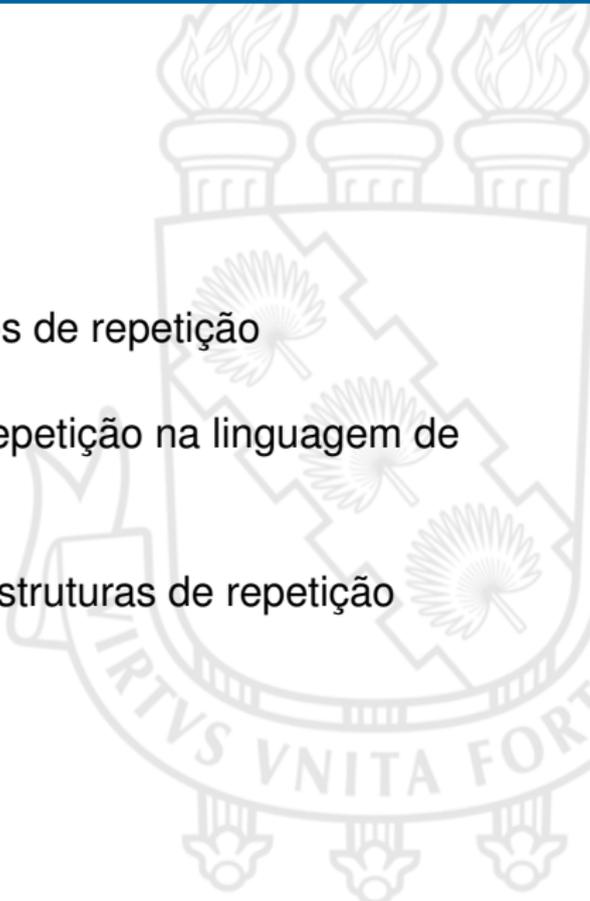
# Sumário

- 1 Objetivos
- 2 Estruturas de Repetição
- 3 Estrutura While
- 4 Estrutura Do-While
- 5 Estrutura For
- 6 Comando Break/Continue
- 7 Goto



# Objetivos

- Aprender quais são os comandos de repetição
- Como utilizar os comandos de repetição na linguagem de programação C
- Fazer exercícios usando estas estruturas de repetição



# Estruturas de Repetição

- Uma estrutura de repetição permite que uma sequência de comandos seja executada repetidamente, **enquanto determinadas condições são satisfeitas**
- Essas condições são apresentadas por expressões lógicas (exemplo:  $A > B$ ;  $C == 3$ ;  $\text{Letra} == 'a'$ )
  - Repetição com Teste no Início
  - Repetição com Teste no Final
  - Repetição com variável de controle

# Estruturas de Repetição

- O real poder dos computadores está na sua habilidade para repetir uma operação ou uma série de operações muitas vezes
- Esta repetição chama-se **laço** (loop) é um dos conceitos básicos da programação estruturada

# Estruturas de Repetição

- Um conjunto de comandos de um algoritmo pode ser repetido quando subordinado a uma condição:

```
enquanto (condição) faça  
comandos;  
fim enquanto
```

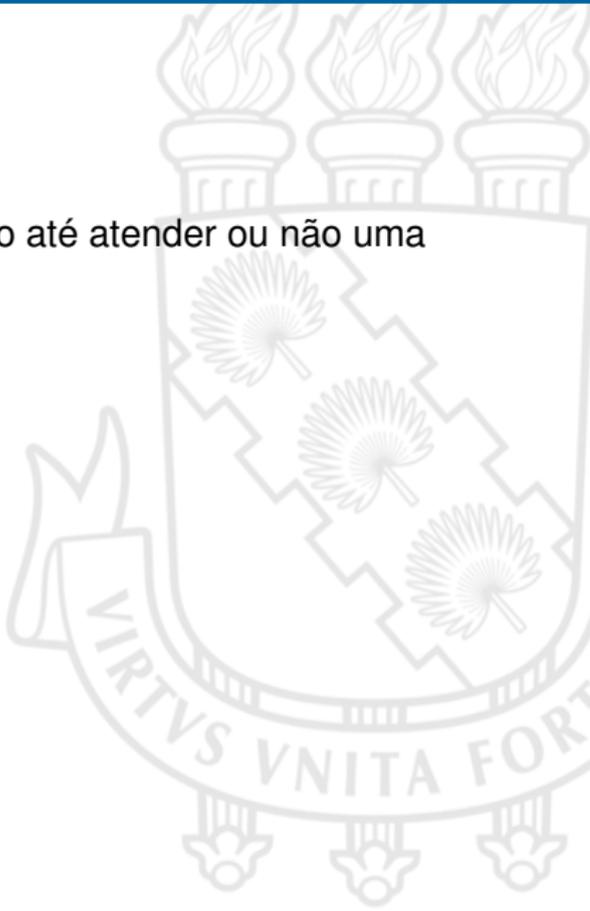
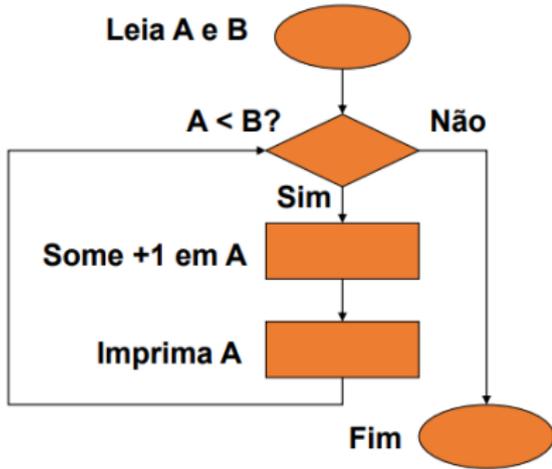
- De acordo com a condição, os comandos serão repetidos zero (se falso) ou mais vezes (enquanto a condição for verdadeira)
  - Essa estrutura normalmente é denominada **laço** ou **loop**

# Estruturas de Repetição

- A condição da cláusula **enquanto** é testada
  - Se ela for verdadeira os comandos seguintes são executados em sequencia como em qualquer algoritmo, até a cláusula **fim-enquanto**
  - O fluxo nesse ponto é desviado de volta para a cláusula **enquanto** e o processo se repete
  - Se a condição for falsa (ou quando finalmente for), o fluxo do algoritmo é desviado para o primeiro comando após a cláusula **fim-enquanto**

# Estruturas de Repetição

- Relembrando em fluxogramas
  - Um processo pode ser repetido até atender ou não uma condição



# Estruturas de Repetição

```
Leia A;  
Leia B;  
Enquanto A > B  
A recebe A + 1;  
Imprime A;  
Fim-Enquanto
```



# Estruturas de Repetição - Loop Infinito

- Um loop ou laço infinito ocorre quando cometemos algum erro
  - Ao especificar a condição lógica que controla a repetição
  - Ou por esquecer de algum comando dentro da iteração

## Condição Errônea

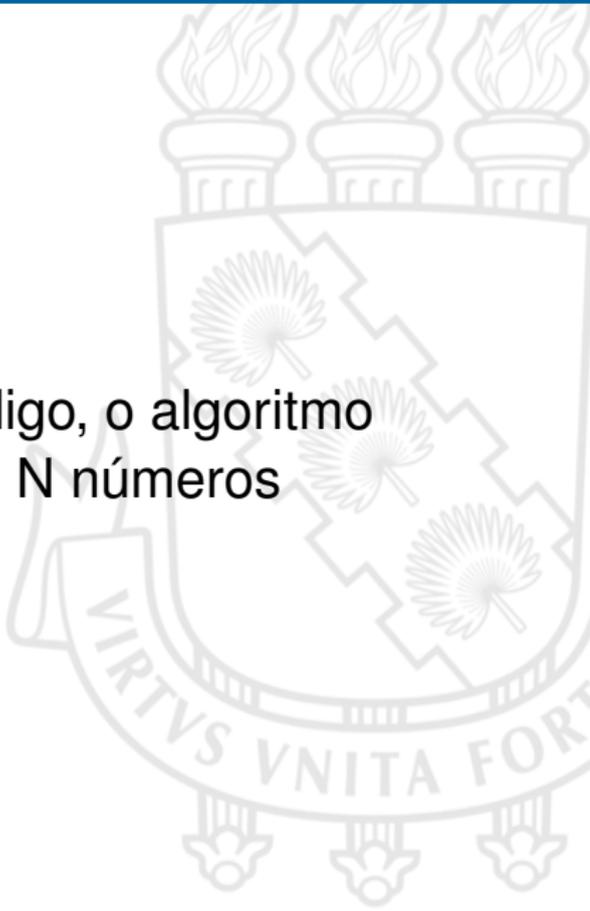
```
x recebe 4;  
Enquanto (x < 5) faça  
x recebe x - 1;  
imprima x;  
fim-enquanto
```

## Não muda valor

```
x recebe 4;  
Enquanto (x < 5) faça  
imprima x;  
fim-enquanto
```

# Exercício 1

- Escreva, em pseudo-código, o algoritmo para calcular a média de N números



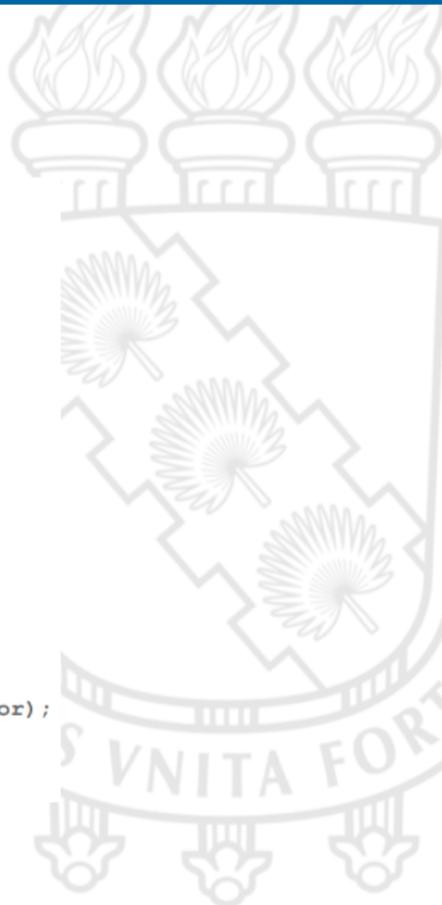
## Exercício 1 - Resolução em Pseudo-código

```
Leia n;  
media recebe 0;  
n1 recebe 0;  
Enquanto (n1 < n):  
  Leia x;  
  media recebe media + x;  
  n1 recebe n1 + 1;  
fim-enquanto  
Imprima media/n;
```

# Exercício 1 - Resolução em C

```
#include<stdio.h>
#include<stdlib.h>
int main(){
    int n, contador = 0;
    float valor, media = 0.0;
    printf("Quantas notas? ");
    scanf("%d", &n);

    while(contador < n){
        printf("Qual a nota? ");
        scanf("%f", &valor);
        media = media + valor;
        contador = contador + 1;
    }
    printf("A media eh: %f\n", media/contador);
    return 0;
}
```



# Estrutura While

- Equivale ao comando “enquanto” utilizado nos pseudo-códigos
  - Repete a sequencia de comandos enquanto a condição for verdadeira
  - **Repetição com Teste no Início**
- Esse comando possui a seguinte forma geral:

```
while (condição) {  
sequencia de comandos;  
}
```

# Estrutura While

- Faça um programa que mostra na tela os números de 1 a 100

```
int main() {  
// o programa mostra na tela os numeros de 1 ate 100  
printf("1 2 3 4 5 6 ...");  
return 0;  
}
```

- A solução acima é inviável para valores grandes. Precisamos de algo mais eficiente e inteligente

# Estrutura While

- Faça um programa que mostra na tela os números de 1 a 100

```
int main () {  
    int numero;  
    numero = 1; Inicializa o contador  
    while (numero <= 100){  
        printf("%d", numero);  
        numero = numero + 1; Incrementa o contador  
    }  
    return 0;  
}
```

- Observe que a variável **numero** é usado como um **contador**, ou seja, vai contar quantas vezes o loop será executado

## Estrutura While - Exemplo 1 (Errado)

- Faça um programa para ler 5 números e mostrar o resultado da soma desses números

```
int main(){  
    float val1, val2, val3, val4, val5, soma;  
  
    printf("\nDigite o 1o. numero: ");  
    scanf("%f", &val1);  
  
    printf("\nDigite o 2o. numero: ");  
    scanf("%f", &val2);  
  
    printf("\nDigite o 3o. numero: ");  
    scanf("%f", &val3);  
  
    printf("\nDigite o 4o. numero: ");  
    scanf("%f", &val4);  
  
    printf("\nDigite o 5o. numero: ");  
    scanf("%f", &val5);  
  
    soma = val1 + val2 + val3 + val4 + val5;  
    printf("\nO resultado da soma eh: %f", soma);  
  
    return 0;  
}
```

# Estrutura While - Exemplo 1 (Correto)

- Faça um programa para ler 5 números e mostrar o resultado da soma desses números

```
int main () {
    float valor, soma;
    int contagem;
    soma = 0; // inicializa o valor de soma
    contador = 1; // inicializa o contador
    while (contagem <= 5){
        printf("\nDigite o valor do numero %d", contador);
        scanf("%f", &valor);
        soma = soma + valor;
        contagem = contagem + 1;
    }
    printf("\nO resultado da soma e: %.2f", soma);
    return 0;
}
```

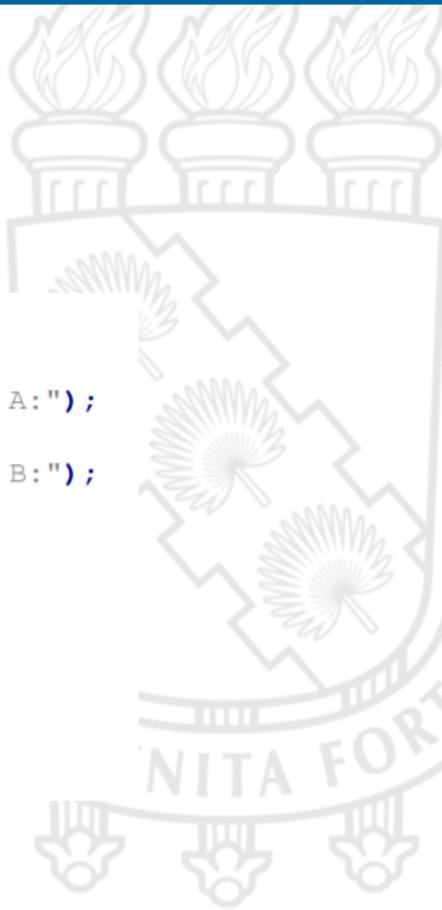
Acumulador

Controla o número de execuções

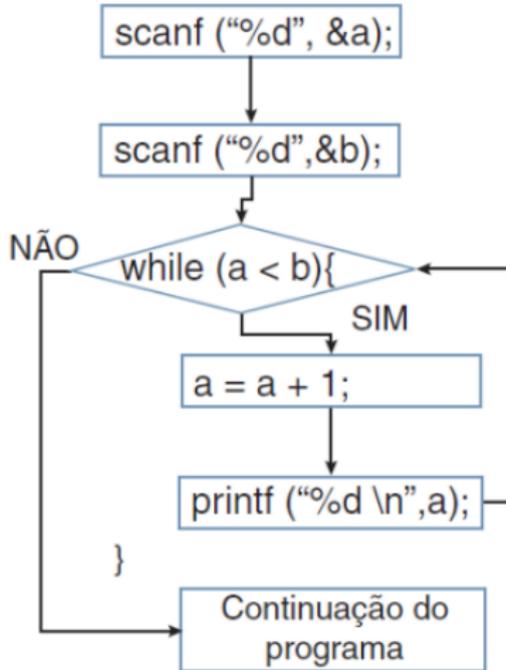
## Estrutura While - Exemplo 2

- Imprimindo os números entre A e B

```
int main() {  
    int a, b;  
    printf("Digite o valor de A:");  
    scanf("%d", &a);  
    printf("Digite o valor de B:");  
    scanf("%d", &b);  
  
    while(a < b){  
        a = a + 1;  
        printf("%d \n", a);  
    }  
    return 0;  
}
```

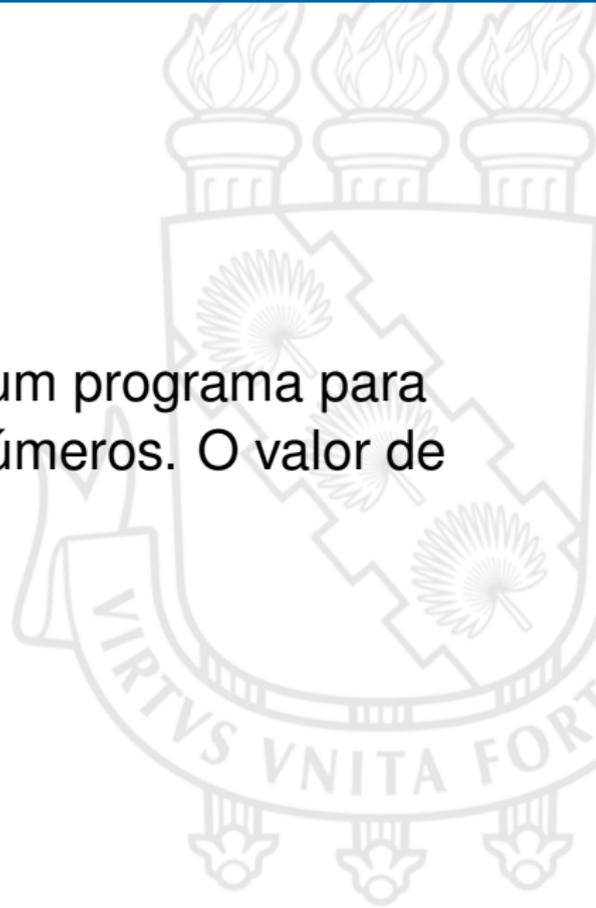


# Estrutura While



## Exercício 2

- Escreva, usando while, um programa para calcular a média de N números. O valor de N é dado pelo usuário.



## Exercício 2 - Resolução

```
int main(){
    int n,n1,x;
    float media = 0;
    printf("Digite N:");
    scanf("%d",&n);
    n1 = 0;
    while (n1 < n){
        printf("Digite X:");
        scanf("%d",&x);
        media = media + x;
        n1 = n1 + 1;
    }
    printf("%f",media/n);

    return 0;
}
```



# Estrutura Do-While

- Comando **while**: é utilizado para repetir um conjunto de comandos zero ou mais vezes
  - Repetição com Teste no Início
- Comando do-while: é utilizado sempre que o bloco de comandos **deve ser executado ao menos uma vez**
  - Repetição com Teste no Final

# Estrutura Do-While

- Executa comandos
- Avalia condição:
  - Se verdadeiro, re-executa o bloco de comandos
  - Caso contrário, termina o laço
- Sua forma geral é (lembre-se: sempre termina com o ponto e vírgula!)

```
do {  
sequencia de comandos;  
} while(condição);
```

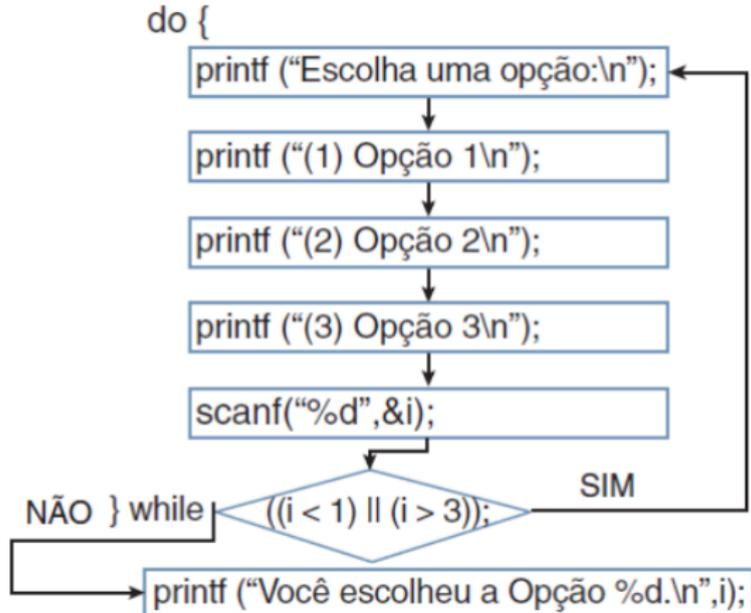
# Estrutura Do-While



```
int main(){
    int i;
    do{
        printf("Escolha uma das opções a seguir:\n");
        printf("(1) opcao 1\n");
        printf("(2) opcao 2\n");
        printf("(3) opcao 3\n");
        scanf("%d", &i);
    } while ((i < 1) || (i > 3));
    system("pause");
    return 0;
}
```



# Estrutura Do-While



# Estrutura For

- O loop ou laço for é usado para repetir um comando, ou bloco de comandos, diversas vezes
  - Maior controle sobre o loop
- Sua forma geral é:

```
for (inicialização; condição; incrementos) {  
sequencia de comandos;  
}
```

# Estrutura For

- **Inicialização:** iniciar variáveis (contador)
- **Condição:** avalia a condição. Se verdadeiro, executa comandos do bloco, senão encerra laço
- **Incremento:** ao término do bloco de comandos, incrementa o valor do contador
  
- Repete o processo até que a condição seja falsa

```
for (inicialização; condição; incrementos) {  
sequencia de comandos;  
}
```

# Estrutura For

- Em geral, utilizamos o comando **for** quando precisamos ir de um valor inicial até um valor final
- Para tanto, utilizamos uma variável para a realizar a contagem
  - Exemplo: `int i;`
- Nas etapas do comando **for**
  - **Inicialização:** atribuímos o valor inicial a variável
  - **Condição:** especifica a condição para continuar no *loop*
    - Exemplo: seu valor final
  - **Incremento:** atualiza o valor da variável usada na contagem

# Estrutura For

- Exemplo: imprime os valores de 1 até 10

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int i;
    for (i = 1; i <= 10; i++) {
        printf("%d\n", i);
    }
    system("pause");
    return 0;
}
```

The diagram highlights the three parts of the for loop: **Inicialização** (i = 1), **Condição** (i <= 10), and **Incremento** (i++).

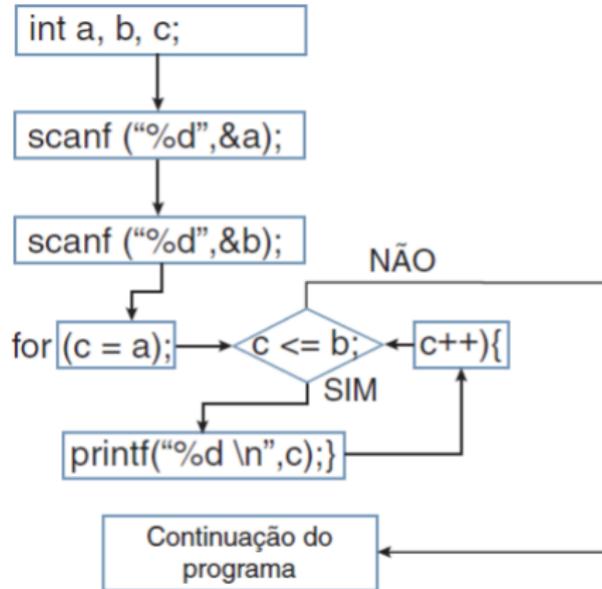
# Estrutura For

- Comando **while**: repete uma sequencia de comandos enquanto uma condição for verdadeiro
- Comandos **for**: repete uma sequencia de comandos “N vezes”
- **Exemplo**:

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int a,b,c;
    printf("Digite o valor de a: ");
    scanf("%d",&a);
    printf("Digite o valor de b: ");
    scanf("%d",&b);
    for(c = a; c <= b; c++){
        printf("%d \n",c);
    }

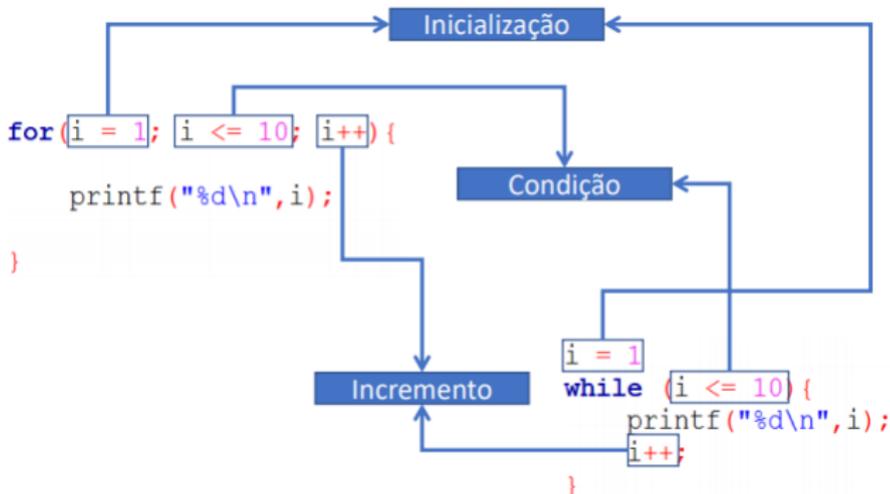
    return 0;
}
```

# Estrutura For



# For vs While

- Exemplo: mostra os valores de 1 até 10



# Estrutura For

- Podemos omitir qualquer um de seus elementos
  - Inicialização, condição ou incremento
- Exemplo: **for** sem inicialização

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int a,b,c;
    printf("Digite o valor de a: ");
    scanf("%d",&a);
    printf("Digite o valor de b: ");
    scanf("%d",&b);
    for (; a <= b; a++){
        printf("%d \n",a);
    }
    system("pause");
    return 0;
}
```

# Estrutura For

- **Cuidado:** for sem condição
  - Omitir a condição cria um laço infinito
  - Condição será sempre verdadeira

```
#include <stdio.h>
#include <stdlib.h>
int main(){
    int a,b,c;
    printf("Digite o valor de a: ");
    scanf("%d",&a);
    printf("Digite o valor de b: ");
    scanf("%d",&b);
    //o comando for abaixo é um laço infinito
    for (c = a; ; c++){
        printf("%d \n",c);
    }
    system("pause");
    return 0;
}
```

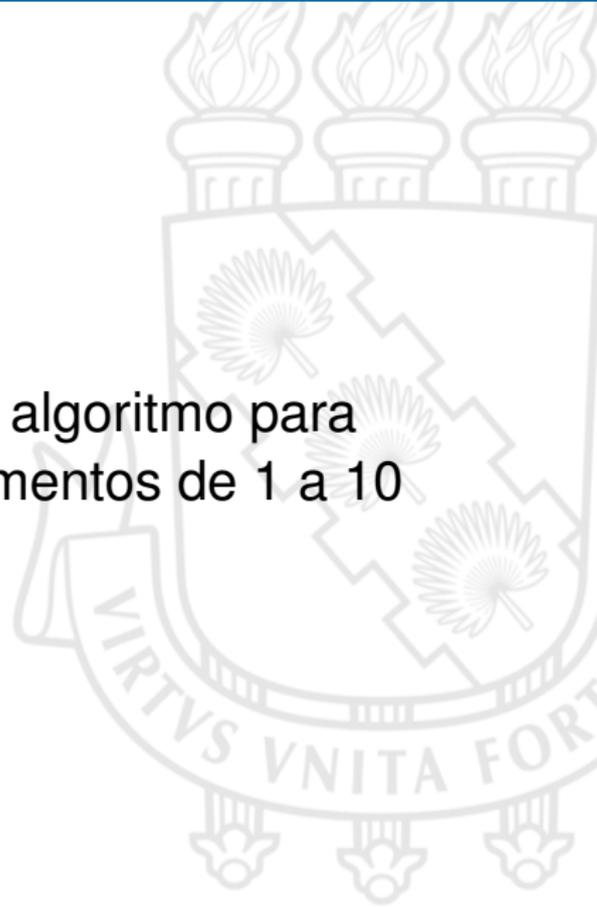
# Estrutura For

- **Cuidado:** for sem incremento
  - Omitir o incremento cria um laço infinito
  - Incremento pode ser feito nos comandos

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int a,b,c;
    printf("Digite o valor de a: ");
    scanf("%d",&a);
    printf("Digite o valor de b: ");
    scanf("%d",&b);
    for (c = a; c <= b; ){
        printf("%d \n",c);
        c++;
    }
    system("pause");
    return 0;
}
```

## Exercício 3

- Escreva, usando for, um algoritmo para calcular a soma dos elementos de 1 a 10



## Exercício 3 - Resolução

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int i, s = 0;
    for(i = 1; i <= 10; i++){
        s = s + i;
    }
    printf("Soma = %d \n", s);
    return 0;
}
```



# Comando Break/Continue

- Já foi visto duas utilizações para o comando break:
  - Interrompendo os comandos switch
  - **Exemplo:**

```
int num;
scanf("%d", &num);
switch(num) {
case 0: printf("zero"); break;
case 1: printf("um"); break;
}
```

# Comando Break/Continue

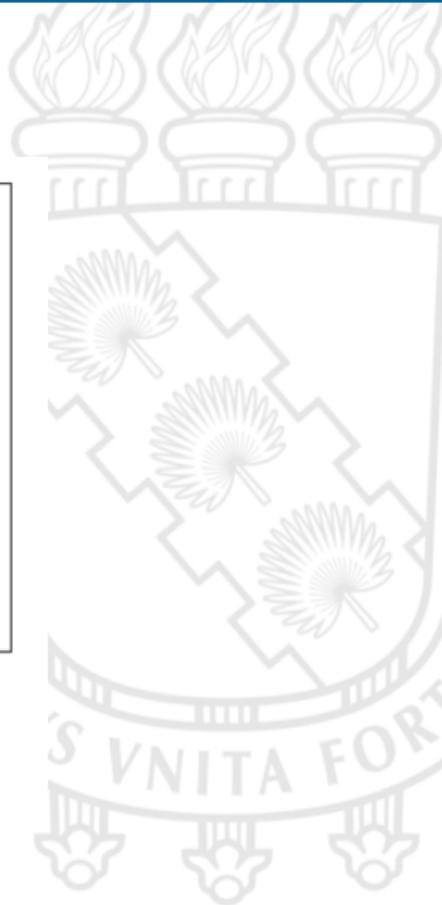
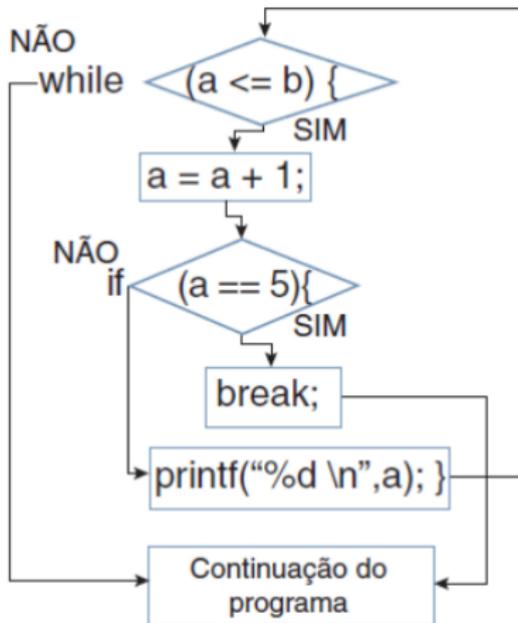
- O comando **break** serve para:
  - Quebrar a execução de um comando (como no caso do **switch**)
  - Interromper a execução de qualquer loop (**for**, **while**, ou **do-while**)
- O comando **break** é utilizado para terminar de forma abrupta uma repetição
  - **Exemplo:** se estivermos dentro de uma repetição e um determinado resultado ocorrer, o programa deverá sair da repetição e continuar na primeira linha seguinte a ela

# Comando Break/Continue

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int a,b;
    printf("Digite o valor de a: ");
    scanf("%d",&a);
    printf("Digite o valor de b: ");
    scanf("%d",&b);
    while (a <= b) {
        a = a + 1;
        if(a == 5)
            break;
        printf("%d \n",a);
    }

    return 0;
}
```

# Comando Break/Continue



# Comando Break/Continue

- Comando **continue**
  - Diferente do comando **break**, somente funciona dentro do loop
  - “Pula” essa interação do loop
  
- Quando o comando continue é executado, os comandos restantes da repetição são ignorados. O programa volta a testar a condição do laço para saber se o mesmo deve ser executado novamente ou não

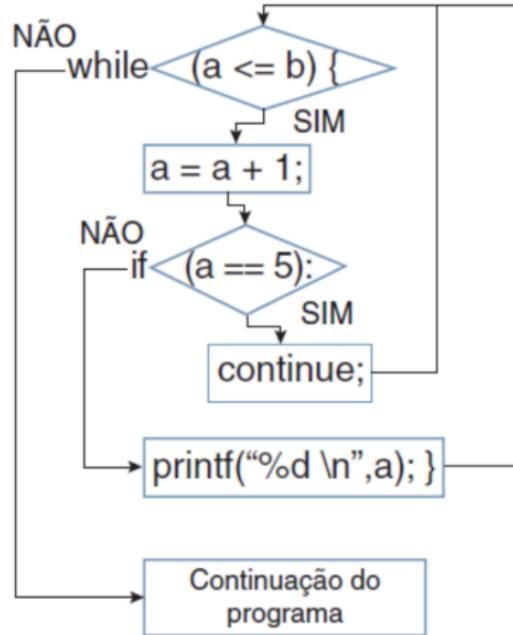
# Comando Break/Continue

```
#include <stdio.h>
#include <stdlib.h>
int main(){
    int a,b;
    printf("Digite o valor de a: ");
    scanf("%d",&a);
    printf("Digite o valor de b: ");
    scanf("%d",&b);
    while (a <= b){
        a = a + 1;
        if(a == 5)
            continue;
        printf("%d \n",a);
    }

    return 0;
}
```



# Comando Break/Continue



# Goto/Label

- É um salto condicional (**goto**) para um local especificado
- Este local é determinado por uma palavra chave no código (**label**)
  - Este local pode ser a frente ou atrás no programa, mas deve ser dentro da mesma função
- Forma geral:

```
palavra_chave:  
goto palavra_chave;
```

## Goto/Label

- O teorema da programação estruturada prova que a instrução goto não é necessária para escrever programas
  - Alguma combinação das três construções de programação (comandos sequenciais, condicionais e de repetição) são suficientes para executar qualquer cálculo
  - Além disso, o uso de goto pode deixar o programa bem ilegível

## Goto/Label

- Apesar de banido da prática de programação, pode ser útil em determinadas circunstâncias
  - **Exemplo:** sair de dentro de laços aninhados

```
#include <stdio.h>
#include <stdlib.h>
int main(){
    int i,j,k;
    for(i = 0; i < 5; i++)
        for(j = 0; j < 5; j++)
            for(k = 0; k < 5; k++)
                if(i == 2 && j == 3 && k == 1)
                    goto fim;
            else
                printf("Posicao [%d,%d,%d]\n",i,j,k);

    fim://label
    printf("Fim do programa\n");

    return 0;
}
```

## Referências

- André Luiz Villar Forbellone, Henri Frederico Eberspächer, **Lógica de programação** (terceira edição), Pearson, 2005, ISBN 9788576050247.
- Ulysses de Oliveira, **Programando em C - Volume I - Fundamentos**, editora Ciência Moderna, 2008, ISBN 9788573936599
- **Slides baseados no material do site “Linguagem C Descomplicado”**
  - <https://programacaodescomplicada.wordpress.com/complementar/>